A Combination of Road Segmentation and Traffic Object Detection

Bingzhao Shan, Songlin Liu, Zihan Wang, Zuoyi Li

1 Executive Overview

To achieve better understanding of autonomous vehicle's surroundings, we proposed a new pipeline (as shown in **Figure.1**) by combining custom variants of SegNet**[1]** for road segmentation and YOLOv1**[2]** for traffic object detection. By utilizing transfer learning and novel image augmentation that is not mentioned in the original configuration, our model obtained better performance on chosen datasets. We were able to achieve good results on both in-domain and out-domain traffic datasets captured in real life.

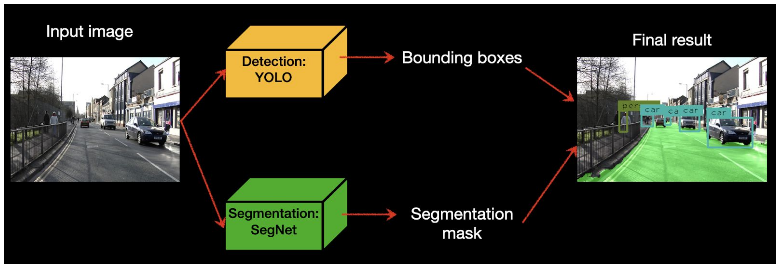


Figure 1: Our pipeline.

2 Background and Impact

Autonomous driving has been a hot topic for a while. Tech giants such as Google, Amazon, and Tesla are all conducting research in this field, not to mention countless researchers exploring the field in universities all around the world. Currently, a core problem of autonomous driving is the detection of different objects such as pedestrians and vehicles in real driving scenarios. Knowledge about road conditions and physical locations of traffic participants from dashcam can provide valuable information to safer self-driving vehicles, so it can stay on the road and stay away from the pedestrians and other vehicles. A good autonomous driving agent can also be provided to human drivers to improve their awareness of traffic related objects in dark environments or complex situations.

3 Method

3.1 Motivation for combining segmentation and detection

Rectangle bounding boxes can be inaccurate when describing the irregular shape of the road due to occlusion and road vanishing, etc. However, segmentation itself is not suitable to describe each individual object we want to avoid such as pedestrians, vehicles. This is because it can be difficult to retrieve the location of each instance of a given class from pixel labels. Following the logic, combining a detection network and a segmentation network becomes a natural choice to address the problem mentioned above. In this project, we used YOLOv1**[2]** as our detection model and SegNet**[1]** as our segmentation model.

3.2 YOLOv1 for detection

YOLOv1 is a convolutional neural network based object detection model. It is extremely fast compared to other top performing models which require a pipeline that involves different components to propose regions of interest, extraction of features, prediction of boxes and post processing. The secret behind YOLO's efficiency is it condenses almost everything mentioned above into a single network. On a high level, the network consists of convolutional blocks which perform feature extraction, and following fully connected layers which predict bounding boxes. Detailed configuration of the architecture can be seen in **Fig 2**. When other models need to train multiple sub-components at training time, YOLO only trains a single neural network. And at test time, when other models are trying to pass results of different components around, YOLO simply runs a forward pass to get the ultimate result we desire.

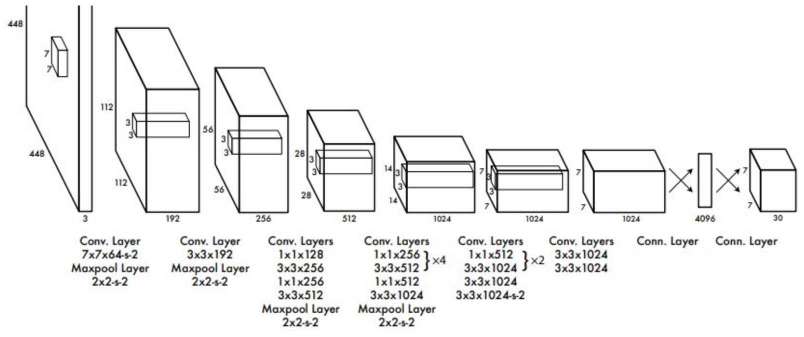

448
7
7
448
3
112
3
3
112
192
56
3
3
56
256
28
3
3
28
512
14
3
3
14
1024
7
3
3
7
1024
7
7
1024
4096
7
7
30
Conv. Layer 7x7x64-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x192 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x128 3x3x256 1x1x256 3x3x512 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x256 3x3x512 ×4 1x1x512 3x3x1024 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x512 3x3x1024 ×2 3x3x1024 3x3x1024-s-2
Conv. Layers 3x3x1024 3x3x1024
Conn. Layer
Conn. Layer

Figure 2: YOLO's network architecture.

3.3 SegNet for segmentation

We used SegNet **[1]** as our segmentation model. SegNet is a convolutional neural network designed for pixel-wise semantic segmentation tasks. Compared to traditional segmentation methods such as graph-cuts **[3]** and some earlier learning-based models such as FCN **[4]**, SegNet is able to learn from samples as well as giving near-real-time predictions on images with fewer trainable parameters. It consists of an encoder and a corresponding decoder. The encoder is identical in shape to the first 13 layers in VGG16 **[9]** and it's supposed to extract high-level features out of the input images. The decoder, on the other hand, is a reverse step of the encoder and it's supposed to generate segmentation masks that are identical to the input images in shape other than the number of channels. After Soft-maxing the final output, we are able to get a segmentation mask from each original image. The architecture is shown in **Figure 3**.

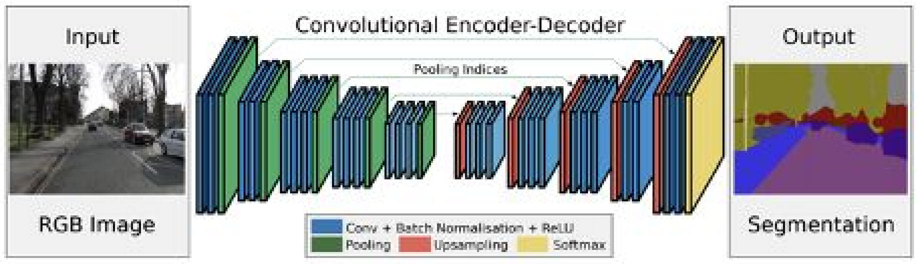


Figure 3: SegNet model architecture cited from the SegNet paper.

4 Prototype

4.1 Packages used imageio, PIL, numpy, PyTorch, torchvision, imgaug, opencv-python

4.2 Novelty

To outperform the benchmark described in the original paper on the CamVid dataset **[6]** as well as giving better out-domain predictions (for example, on the KITTI dataset **[7]**), we adapted some strategies for SegNet such as data augmentation, learning weight decay, transfer learning, etc. We also used similar mesure on YOLO.

4.3 YOLOv1

Except for the notes mentioned above, our YOLOv1 variant followed most of the configuration in the paper **[2]**. The labels for training images (bounding boxes and their classes) are transformed into w.r.t the cells that contain the bounding box's center, instead of w.r.t the whole image. In the forward pass, instead of using the feature extraction network in **Fig 2**, we replaced it with ResNet152[8] pre-trained by Torchvision for better weight initialization. After the forward pass of batch size 16, we compute the loss which consists of bounding box coordinates and width and height error, object presence error and class error. The loss is used in backward pass to update the weights and bias in different layers. Another thing to mention is that instead of SGD, we used Adam with learning rate 1e-5 for optimization. During test time, before we overlay bounding boxes onto the original image, we first perform non-maximal suppression of threshold 0.4 to get rid of overlapping boxes and low confidence boxes.

4.4 SegNet

The implementation of SegNet is also based on the configuration described in the original SegNet paper using PyTorch. Since the road is the only class we intended to segment in this project, we labeled all other classes as background. To achieve lower loss, we decayed our learning rate after a certain number of epochs. And to make a fair comparison between the model trained directly on CamVid as described in the paper and the model we improved in the project, we used the same optimizer (Adam, lr=0.1, with its learning rate divided by 3 every 15 epochs) and the same loss function (Cross-entropy loss with median frequency balancer as weights) all across our training. Note that all training is done on CamVid's training dataset. For initialization, we tried to initialize the encoder with and without VGG16. We also applied online augmentation in our training (details can be found in **Figure 4**).

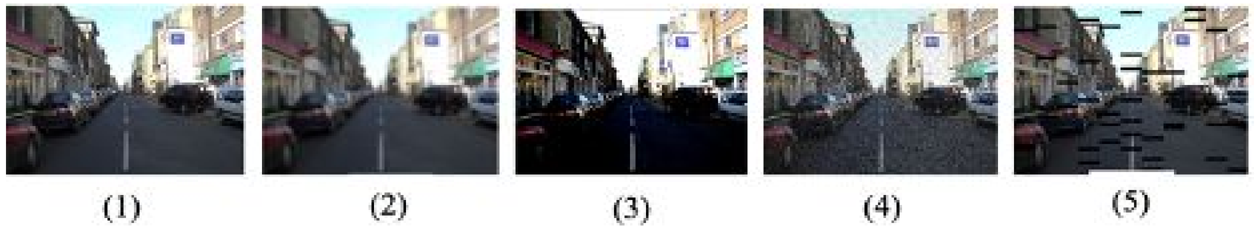


Figure 4. Image augmentation effects applied to the training set. We amplified results for our demonstration. Pictures from left to right are (1) The original image (2) Blur (3) Contract and intensity change (4) Noise (5) Random patch removed (to simulate shadow). Each of these effects is applied to the original image with a certain probability and in random order.

5 Results and Future work

We evaluated our two modules separately and combined them together for end-to-end visualization. The two datasets we used here are the CamVid test dataset (as our in-domain evaluation dataset, quantitatively) and the KiTTi dataset(as our out-of-domain evaluation dataset visually). Images are cropped or resized to be 360x480. Visual results are shown in **Figure 5**.

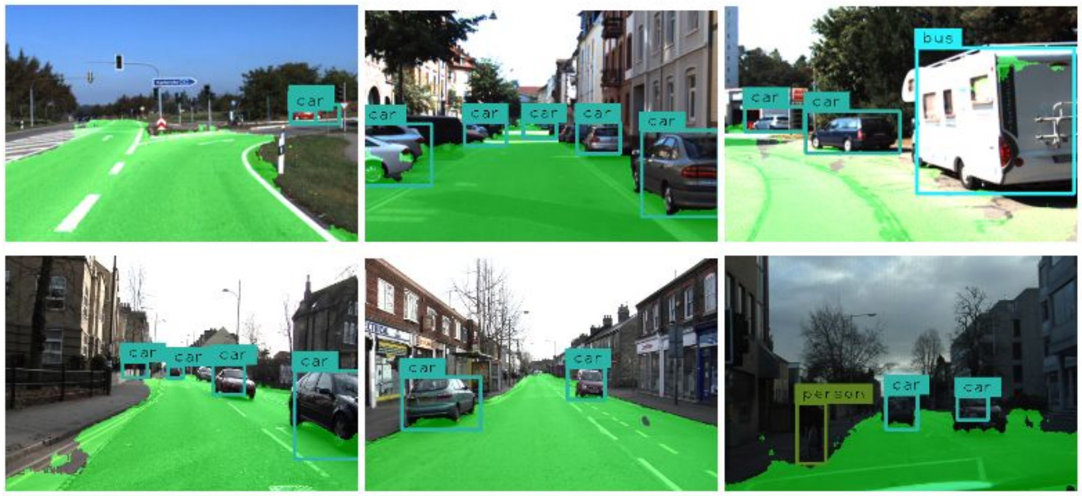


Figure 5. YOLOv1 and SegNet correct test results. Upper three examples are in-domain CamVid examples. Lower three examples are out-domain KiTTi examples.

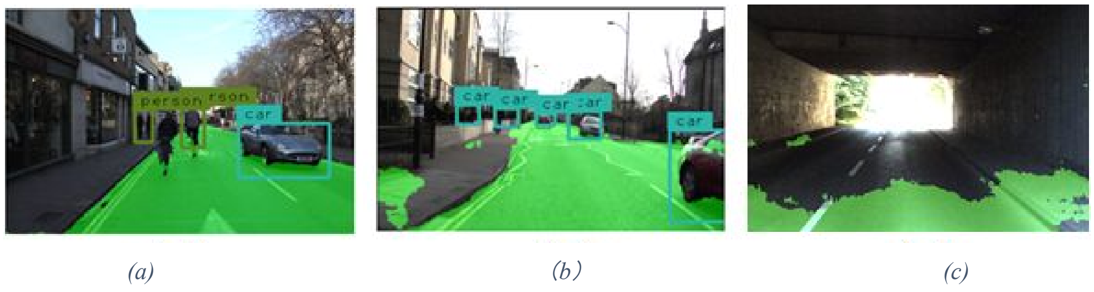
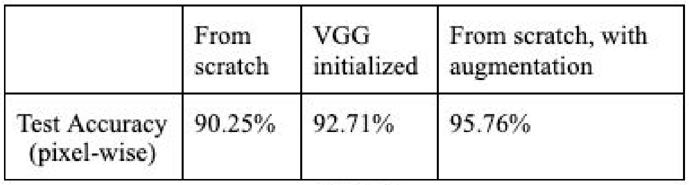


Figure 6: examples of YOLOv1 and SegNet poor performance

Our implementation of YOLOv1 was able to achieve 52.13% mean average precision on PASCAL VOC [5] 2012's validation set within 10 hours of training. Some examples of failures are shown in **Figure. 6 (a), (b)**. We can see that the YOLOv1 model may misclassify an object, or miss some objects. We argue that with more exhaustive training of the model, we can address these problems and achieve higher accuracy. We can also implement the YOLOv5 model to get better performance.

Table 1. SegNet Test Accuracy



	From scratch	VGG initialized	From scratch, with augmentation
Test Accuracy (pixel-wise)	90.25%	92.71%	95.76%

For SegNet, the quantitative evaluation results on the CamVid test dataset is shown in **Table1**. We also visualized some bad segmentation results obtained from our best model (trained from scratch, with augmentation) and a typical one can be found in **Figure 6 (c)**. The figure indicates that our model can be improved on images with heavy shadow or low intensity levels. Future work on SegNet include: (1) Introduce new loss functions such as focal loss **[10]** or some boundary loss to handle data imbalance more efficiently and reinforce boundary smoothness. (2) Find a more effective way to simulate shadow on our dataset while doing data augmentation.

References

[1] Badrinarayanan, Vijay, Kendall, Alex, and Cipolla, Roberto, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," 2015.

[2] Redmon, Joseph, Divvala, Santosh, Girshick, Ross, and Farhadi, Ali, "You Only Look Once: Unified, Real-Time Object Detection," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, vol. 2016-, pp. 779–788.

[3] Boykov, Y.Y and Jolly, M.-P, "Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images," in Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001, 2001, vol. 1, pp. 105–112 vol.1.

[4] Long, Jonathan, Shelhamer, Evan, and Darrell, Trevor, "Fully Convolutional Networks for Semantic Segmentation," 2014.

[5] PASCAL VOC. http://host.robots.ox.ac.uk/pascal/VOC/ Accessed: Dec 14th, 2020

[6] Brostow, Gabriel J, Fauqueur, Julien, and Cipolla, Roberto, "Semantic object classes in video: A high-definition ground truth database," Pattern recognition letters, vol. 30, no. 2, pp. 88–97, 2009.

[7] Geiger, A, Lenz, P, and Urtasun, R, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in 2012 IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 3354–3361.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep Residual Learning for Image Recognition," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, vol. 2016-, pp. 770–778.

[9] Simonyan, Karen and Zisserman, Andrew, "Very Deep Convolutional Networks for Large-Scale Image Recognition," 2014.

[10] Tsung-Yi Lin, Goyal, Priya, Girshick, Ross, Kaiming He, and Dollar, Piotr, "Focal Loss for Dense Object Detection," in 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2999–3007.

Git Repo

https://github.com/zuoyigehaobing/LaneUnderstanding